(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86630 A2

(51) International Patent Classification[7]: G10K

(21) International Application Number: PCT/GB01/02021

(22) International Filing Date: 4 May 2001 (04.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

0010967.8	5 May 2000 (05.05.2000)	GB
0010969.4	5 May 2000 (05.05.2000)	GB
0011178.1	9 May 2000 (09.05.2000)	GB
0022164.8	11 September 2000 (11.09.2000)	GB
0030843.7	18 December 2000 (18.12.2000)	GB

(71) Applicant *(for all designated States except US)*: **SSEYO LIMITED** [GB/GB]; Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

(72) Inventors; and

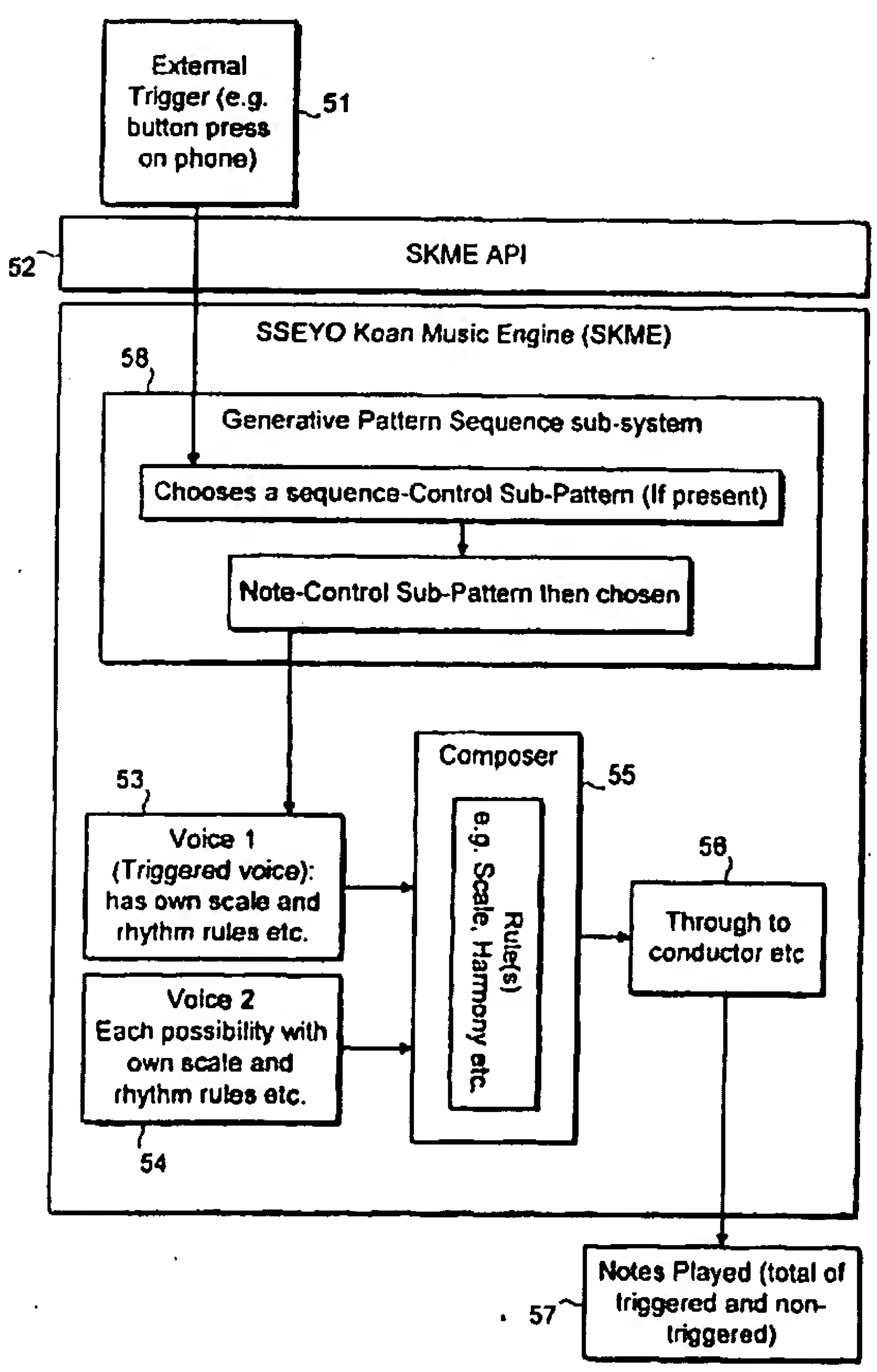
(75) Inventors/Applicants *(for US only)*: **COLE, John, Tim** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **COLE, Murray, Peter** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

(74) Agents: **MAGGS, Michael, Norman** et al.; Kilburn & Strode, 20 Red Lion Street, London WC1R 4PJ (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: AUTOMATED GENERATION OF SOUND SEQUENCES



(57) Abstract: A generative audio system has an audio or music engine for generatively creating an audio work. While the work is in the process of being created, the engine receives an external trigger (51) - for example the button-press on a mobile phone - and the engine then controls or influences the work in progress in dependence upon that trigger. In one embodiment, the external trigger may control or select a new musical voice which harmonizes with the existing voices as creation of the work continues.

WO 01/86630 A2

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Automated Generation of Sound Sequences

This invention relates to methods and systems for automated generation of sound sequences, and especially (though not exclusively) of sound sequences in the form of music.

The automated creation of music has a long history, going back at least as far as Mozart's use of musical dice. One of the first musical works generated by a computer was L. Hiller's Illiac suite. Since that time, of course, the sophistication of computer-generated music or more generally audio sequences has increased substantially.

Systems for creating musical sequences by computer may conveniently be divided up into two areas, which have been called "non-generative" and "generative". Non-generative systems include deterministic systems which will produce the same sequences every time, along with systems that simply replay (perhaps in a random or other order) pre-composed sections of music. The vast majority of current systems which produce musical output make use of this type of approach, for example by selecting and playing a particular predefined sequence of notes at random when the key is pressed or a mouse button clicked. Generative Music Systems, on the other hand, may be considerably more complex. Such systems generate musical content, typically note by note, on the basis of a higher-level of musical knowledge. Such systems either explicitly or implicitly are aware of a variety of musical rules which are used to control or influence the generation of the music. In some systems, the rules may operate purely on the individual notes being generated, without imposing any form of higher order musical structure on the output; in such systems, any musical order that arises will be of an emergent nature. More sophisticated systems may include higher-level rules which can influence the overall musical structure. Generative Music Systems will normally create

musical content "on the fly", in other words the musical sequences are built up note by note and phrase by phrase, starting at the beginning and finishing at the end. This means that – in contrast with some of the non-generative systems - the musical content can be generated and played in real time: there is no need for example for the whole of the phrase to be generated before the first few notes of the phrase can be played.

For our present purposes, the essential features of a generative music system are that it generates musical content in a non-deterministic way, based upon a plurality of musical rules (which may either be implicit within the software or which may be explicitly specified by either the program writer or the user of the program). By analogy, a generative sound system produces non-deterministic sound sequences based upon sound-generation rules.

According to the present invention there is provided a generative audio system including an audio engine for generatively creating an audio work and, while the work is being created, receiving a trigger from an external event and controlling or influencing the work in progress in dependence upon the trigger.

More generally, there may be provided a method or a system in which composition of a sequence of sounds dynamically is triggered by an external, arbitrarily-timed event.

The method and system of the invention enable the composition in real-time of complex sequences of music and audio events to be achieved in response to external stimuli. A generative music process of the method or system may respond to the external triggering to produce generative pattern sequences that contribute to the music output. The process may ensure that the triggered music event is co-ordinated with any currently-playing audio environment.

A method and system for automated generation of sound sequences according to a preferred embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a schematic representation of the preferred system of the invention;

Figure 2 is illustrative of objects that are involved in a component of the system of Figure 1;

Figure 3 is a flow-chart showing process steps involved in control sequencing within the method and system of the invention;

Figure 4 is illustrative of operation of the method and system of the invention in relation to scale and harmony rules; and

Figure 5 illustrates operation of the method and system of the invention in relation to the triggering of note sequences and their integration into a musical work as currently being composed and played.

The method and system to be described are for automated generation of sound sequences and to integrate data presented or interpreted in a musical context for generating an output reflecting this integration. Operation is within the context of generation of musical works, audio, sounds and sound environments in real-time. More especially, the method and system function in the manner of a `generative music system' operating in real-time to enable user-interaction to be incorporated into the composition on-the-fly. The overall construction of the system is shown in Figure 1 and will now be described.

Referring to Figure 1, the system involves four high-level layers, namely, an applications layer I comprising software components 1 to 5, a layer II formed by

an application programmer's interface (API) 6 for interfacing with a music engine SKME that is manifest in objects or components 7 to 14 of a layer III, and a hardware device layer IV comprising hardware components 15 to 19 that interact with the music engine SKME of layer III. Information flow between the software and hardware components of layers I to IV is represented in Figure 1 by arrow-heads on dotted-line interconnections, whereas arrow-heads on solid lines indicate an act of creation; for example, information in the composed-notes buffer 11 is used by the conductor 12 which is created by the soundscape 8.

The applications layer I determines the look, feel and physical instantiation of the music engine SKME. Users can interact with the music engine SKME through web applications 1, or through desktop computer applications 2 such as those marketed by the Applicants under their Registered Trade Mark KOAN as KOAN PRO and KOAN X; the music engine SKME may itself be such as marketed by the Applicants under the Registered Trade Mark KOAN. Interaction with the engine SKME may also be through applications on other diverse platforms 3 such as, for example through mobile telephones or electronic toys. All applications 1 to 3 ultimately communicate with the music engine SKME via the API 6 which protects the internals of the music engine SKME from the outside world and controls the way in which the applications can interact with it. Typically, the instructions sent to the API 6 from the applications 1 to 3 consist of commands that instruct the music engine SKME to carry out certain tasks, for example starting the composition and playback, and changing the settings of certain parameters (which may affect the way in which the music is composed/played). Depending on the needs of the individual applications, communication with the API 6 may be direct or via an intermediate API. In the present case communication to the API 6 is direct from the desktop computer applications 2, whereas it is via an intermediate browser plug-in API 4 and Java API 5 from applications 1 and 3 respectively.

The music engine SKME, which is held in memory within the system, comprises eight main components 7 to 14. Of these, SSFIO 7, which is for file input/output, holds a description of the parameters, rules and their settings used by algorithms within the engine, to compose. When the engine SKME is instructed via the API 6 to start composition/playback, a soundscape 8 is created in memory and this is responsible for creating a composer 10, conductor 12 and all the individual compositional objects 9 relating to the description of the piece as recorded in the SSFIO 7. The compositional objects are referred to by the composer 10 to decide what notes to compose next. The composed notes are stored in a number of buffers 11 along with a time-stamp which specifies when they should be played. The conductor 12 keeps time, by receiving accurate time information from a timer device 19 of level IV. When the current time exceeds the time-stamp of notes in the buffers 11, the relevant notes are removed from the buffers 11 and the information they contain (such as concerning pitch, amplitude, play time, the instrument to be used, etc.) is passed to the appropriate rendering objects 13. The rendering objects 13 determine *how* to play this information, in particular whether via a MIDI output device 17, or as an audio sample via an audio-out device 18, or via a synthesiser engine 14 which generates complex wave-forms for audio output directly, adding effects as needed.

The hardware devices layer IV includes in addition to the devices 17 to 19, a file system 15 that stores complete descriptions of rules and parameters used for individual compose/playback sessions in the system; each of these descriptions is stored as an `SSfile', and many of these files may be stored by the file system 15. In addition, a MIDI in device 16 is included in layer IV to allow note and other musical-event information triggered by an external hardware object (such as a musical keyboard) to be passed into the music engine SKME and influence the composition in progress.

The system can be described as having essentially two operative states, one, a

`dynamic' state, in which it is composing and the other, a `static' state, in which it is not composing. In the static state the system allows modification of the rules that are used by the algorithms to later compose and play music, and keeps a record encapsulated in the SSFIO component 7, of various objects that are pertinent to the description of how the system may compose musical works. The system is also operative in the dynamic state to keeps records of extra objects which hold information pertinent to the real-time composition and generation of these works. Many of these objects (the compositional objects 9 for example) are actual instantiations in memory of the descriptions contained in the SSFIO 7. Modification of the descriptions in the SSFIO 7 via the API layer II during the dynamic state, results in those modifications being passed down to the compositional objects 9 so that the real-time composition changes accordingly.

Figure 2 shows a breakdown of the SSFIO component 7 into its constituent component objects which exist when the system is in its static and dynamic states; the system creates real-time versions of these objects when composing and playing. In this respect, SSfiles 20 stored each provide information as to `SSObject(s)' 21 representing the different *types* of object that can be present in the description of a work; these objects may, for example, relate to piece, voice, scale rule, harmony rule, rhythm rule. Each of these objects has a list of `SSFparameters' 22 that describe it; for example, they may relate to tempo, instrument and scale root. When an SSfile 20 is loaded into the music engine SKME, actual instances of these objects 21 and their parameters 22 are created giving rise to `SSFObjectInstance' 23 and `SSFParameterInstance' 24 as illustrated in Figure 2.

Referring again to Figure 1, the user interacts with the system through applications 1 to 3 utilising the services of the API 6. The API 6 allows a number of functions to be effected such as `start composing and playing', `change the rules used in the composition', `change the parameters that control how the piece

is played' including the configuration of effects etc. One of the important aspects of the method and system of the invention is the ability to trigger generative pattern sequences in response to external events. The triggering of a generative pattern sequence has a range of possible outcomes that are defined by the pattern sequence itself. In the event that a generative pattern sequence is already in operation when another trigger event is received, the currently operational sequence is ended and the new one scheduled to start at the nearest availability.

Generative pattern sequences allow a variety of musical seed phrases of any length to be used in a piece, around which the music engine SKME can compose in real time as illustrated in Figure 3. More particularly, the generative pattern sequence contains a collection of one or more note-control sub-patterns with or without one or more additional sequence-control sub-patterns.

Three types of note-control sub-patterns can be created, namely: `rhythm' note-control sub-pattern containing note duration information, but not assigning specific frequencies to use for each note; `frequency and rhythm' note-control sub-pattern containing both note duration and some guidance to the generative music engine SKME as to the frequency to use for each note; and `forced frequency' note-control sub-pattern containing note duration, temporal positioning and explicit frequency information to use for each note. Each note-control sub-pattern may also specify ranges of velocities and other musical information to be used in playing each note.

Sequence-control sub-patterns, on the other hand, can be used to specify the sequence in which the note-control sub-patterns are played. The music engine SKME allows the use of multiple sub-patterns in any generative pattern sequence.

To take a particular example, a sequence-control sub-pattern might, for instance,

instruct the system to play the first note-control sub-pattern once, the second note-control sub-pattern twice, then select the third or fourth pattern at random, and play that twice. Then, repeat. Because of the generative nature of the system it will be understood, of course, that repeating a note-control sub-pattern will not necessarily reproduce the same notes as before (unless the note-control sub-pattern is of the "forced frequency" type).

Once one sequence-control sub-pattern has been completed, the next is selected at random (according to user-defined waitings if appropriate), and that pattern is then executed.

Referring more specifically to Figure 3, the step 30 of triggering the generative pattern sequence acts through step 31 to determine whether there are any other relevant sequence-control sub-patterns. If not, a note-control sub-pattern is chosen at random in step 32 from a defined set; each note-control sub-pattern of this set may be assigned a value that determines its relative probability of being chosen. Once it is determined in step 33 that the selected note-control sub-pattern is finished, another (or the same) note-control sub-pattern is selected similarly from the set. The generative pattern sequence continues to play in this manner until instructed otherwise.

If the result of step 31 indicates that there is one or more sequence-control sub-patterns operative, then any sequence-control sub-pattern is chosen at random in step 34 from the defined set; each sequence-control sub-pattern may be assigned a value that determines its relative probability of being chosen. Once a sequence-control sub-pattern has been selected in step 34, it is consulted to determine in step 35 a sequence of one or more note-control sub-patterns to play. As each note-control sub-pattern comes to an end, step 36 prompts a decision in step 37 as to whether each and every specified note-control sub-pattern of the operative sequence has played for the appropriate number of times. If the answer is NO,

then the next note-control sub-pattern is brought into operation through step 35, whereas if the answer is YES another, or the same, sequence-control sub-pattern is selected through repetition of step 34. As before, the generative pattern sequence continues to play in this manner until instructed otherwise.

Each sequence-control sub-pattern defines the note-control sub-pattern(s) to be selected in an ordered list, where each entry in the list is given a combination of: (a) a specific note-control sub-pattern to play, or a range of note-control sub-patterns from which the one to play is chosen according to a relative probability weighting; and (b) a value which defines the number of times to repeat the selected note-control sub-pattern, before the next sequence-control sub-pattern is selected. The number of repetitions may be defined as a fixed value (e.g. 1), as a range of values (e.g. repeat between 2 and 5 times), or as a special value indicating that the specified note-control sub-pattern should be repeated continuously.

Depending upon the note-control sub-pattern operational at any moment after a generative pattern sequence is triggered, various rules internal to the music engine SKME may be used to determine the exact pitch, duration and temporal position of the notes to be played. For example, if a `rhythm' note-control sub-pattern is in operation at a particular point in the generative pattern sequence, then the scale rule, harmony rule and next-note rule within the music engine SKME for that `triggered voice' will be consulted to obtain the exact notes. Alternatively, if the `forced frequency' note-control sub-pattern is operational, no internal rules need be consulted since all the note information is already specified. Furthermore, for the case of `frequency and rhythm', the music engine SKME combines the given frequency offset information with its rules and other critical information such as the root of the current scale and range of available pitch values for the voice in question.

The rules and other parameters affecting composition (e.g. tempo) within the music engine SKME are defined in memory, specifically within the SSFIO 7, and its real-time instantiation of the compositional objects 9. Use of rules and parameters within the music engine SKME form part of the continual compositional process for other voice objects within the system. Figure 4 illustrates this more general process based on examples of scale and harmony rules shown at (1) and (2) respectively.

Referring to Figure 4, the scale rule is illustrated at (1) with shaded blocks indicating a non-zero probability of choosing that interval offset from a designated scale root note. The larger the shaded block, the greater the probability of the system choosing that offset. Thus, for this example, the octave Ove, major third M3 and fifth 5 are the most likely choices, followed by M2, 4, M6 and M7; the rest will never be chosen. Sequences that may be generated by the system from this are shown below the blocks, and in this respect the octave has been chosen most often followed by the major third and the fifth. With the scale root set in the system as C, the resulting sequence of notes output from the system in this example are C,E,C,D,G,A,E,D,C,G,E,B,C,F, as illustrated at (1) of Figure 4.

The harmony rule defines how the system may choose the pitches of notes when other notes are playing, that is to say, how those pitches should harmonise together. In the example illustrated at (2) of Figure 4, only the octave and major second are indicated (by shading) to be selected. This means that when the pitch for a voice is chosen, it must be either the same pitch as, or a major second from, all other notes currently being played.

For the purpose of further explanation, consideration will be given to the example represented at (3) of Figure 4 involving three voice objects V1-V3. The rhythm rules applicable to the voice objects V1-V3 in this example, give rise to a

generated sequence of notes as follows: voice V1 starts playing a note, then voice V2 starts playing a note, then voice V3 starts playing a note, and then after all notes have ended, voice V2 starts playing another note, followed by voice V1 and then voice V3. With this scenario, the note from voice V2 must harmonise with that of voice V1 and the voice V3 note must harmonise with that of voice V2. If in these circumstances the voice V1 is, as illustrated by bold hatching, chosen with a pitch offset of a fifth from the scale root, the pitch for voice V2 must either be the same as (Ove) or a major second above (M2) the fifth. In the case illustrated, it is chosen to be the same, and so the fifth is chosen too. When voice V3 starts playing it must harmonise with both voices V1 and V2, so the pitch chosen must be the same as, or a major second above that of voices V1 and V2. As illustrated, the system chooses voice V3 to be a major second above, therefore giving pitch offset M6 from the scale root.

After voice V3 all notes end, and the next note begins, as illustrated at (4) of Figure 4 with voice V2. This next note by voice V2 is governed by the next-note rule used by voice V2, and the last note played by voice V2. According to this rule, the system chooses pitch offset M2 for voice V2, and then harmonises voices V3 and V1 with it by choice of a major second for both of them. With the scale root set in the system to C, the entire generated sequence accordingly follows that indicated at (5) of Figure 4, where `S' denotes a note starting and `E' a note ending.

Thus, when sequences are generated in response to an external trigger, the actual pitches and harmonisation of that sequence is determined by the composer 10 using several items of information, namely: (a) the note-control sub-pattern operational at that moment; (b) the scale, rhythm, harmony and next-note rules depending upon the type of the note-control subsequence; and (c) any piece-level rules which take into account the behaviour of other voices within the piece.

The external trigger may be user-generated, and may vary according to the particular application. When the generative music system is contained within a mobile phone, the external trigger could be generated by one of the phone buttons being pressed. In other embodiments, the trigger could be generated by some other physical event such as a person walking into a room, a door being opened or closed, a key being turned, a device being powered-up or switching from one node to another, a sound (e.g. speech) event, or indeed any other type of physical event. External triggers could also be generated automatically and/or randomly, for example in response to the ambient temperature, weather, lighting, humidity, infra-red or the like. The trigger could also be produced when an external signal or message (e.g. an e-mail) is received: such an approach might be particularly useful in a mobile phone environment. The trigger could depend upon the message content – e.g. it would be dependent upon the content of an e-mail reporting stock-market figures. In a computer environment – for example within a computer game – the external trigger could represent a keystroke press or mouse click, a pen event, a mouseover event, a screen-touch event, or any other computer-related event either generated by the user or automatically generated by the computer system itself.

When the music engine SKME is in dynamic (i.e. composing and playing) mode, it typically contains a number of voice compositional objects 9. The composer 12 composes a sequence of notes for each of these and makes sure they obey the various rules. The process involved is illustrated in the flow diagram of Figure 5.

Referring to Figure 5, the music engine SKME responds to an external trigger applied at step 51, and the API 6 through step 52 instructs a voice 1 in step 53 to register that it must start a sequence. Voice 1 and the voices 2 to N in step 54, have their own rules, and the composer 10 ensures that the relevant rules are obeyed when utilising any of the voices 1 to N. More particularly, the composer 10 responds in step 55 to the instruction of step 53 for voice 1 to start a sequence,

by starting the generative pattern sequence sub-system of Figure 3. This sends note-control sub-sequences to the trigger voice (voice 1 in this example), but the composer 10 makes sure the resulting notes harmonise with the other voices in the piece. The outcome via the conductor 12 in step 56 is played in step 57.

The generative pattern sequence triggered will play forever, or until the system is instructed otherwise. If a sequence control sub-pattern is used to define a generative pattern sequence such that the final note control sub-pattern is one which plays silence (rest notes) in an infinite loop, then when this pattern sequence is selected, the voice will become effectively `inactive' until another trigger is detected. Further triggering events for the same generative pattern sequence may sound different as the process is generative, or since the rules in use by the piece or the scale of the trigger voice, its harmony or next note rules may have changed (either via interaction through the API 6 or via internal music engine SKME changes).

The sounds used to `render' each note, whether from triggered sequences or generative voices may be played either through the MIDI sounds or the samples of the rendering objects 13, or via software of the synthesiser engine 14 which may add digital signal processing effects such as, for example, filter sweeps, reverberation and chorus. The entire process can be used to generate musical event information that is then fed into, and may thus control, other processing units within the system such as synthesiser related units allowing the triggering of generative sound effects. Voices can also be added which make use of the software synthesiser engine 14 to generate non note-based effects such as sound washes and ambient environmental sounds, such as chimes, wind and other organic sounds.

CLAIMS:

1. A generative audio system including an audio engine for generatively creating an audio work and, while the work is being created, receiving a trigger from an external event and controlling or influencing the work in progress in dependence upon the trigger.

2. A generative audio system as claimed in claim 1 in which the work is created and played in real time.

3. A generative audio system as claimed in claim 1 or claim 2 in which the trigger controls a musical event which is co-ordinated with the work in progress.

4. A generative audio system as claimed in claim 3 in which the work in progress consists of a plurality of individual existing voices, the trigger effecting the addition of a new voice which is co-ordinated with the existing voices as creation of the work continues.

5. A generative audio system as claimed in claim 4 in which the new voice harmonizes with the existing voices.

6. A generative audio system as claimed in any one of the preceding claims in which the audio engine comprises a music engine.

7. A generative audio system as claimed in claim 6 in which the audio engine creates the work in accordance with a plurality of note-control patterns, the note-control patterns defining at least some of the musical rules by which the engine generates musical notes.

8. A generative audio system as claimed in claim 7 in which the trigger

effects the selection of a note-control pattern.

9. A generative audio system as claimed in claim 7 or claim 8 in which the note-control patterns are used by the music engine in a sequence defined by a sequence-control pattern.

10. A generative audio system as claimed in claim 9 in which the trigger effects the selection of a sequence-control pattern.

11. A generative audio system as claimed in any one of the preceding claims in which the trigger is user-generated.

12. A generative audio system as claimed in any one of claims 1 to 10 in which the trigger is automatically generated.

13. A generative audio system as claimed in any one of claims 1 to 10 in which the trigger is representative of a button-press, a keystroke, or a touch of a computer screen.

14. A generative audio system as claimed in any one of claims 1 to 10 in which the trigger is representative of a computer mouse event.

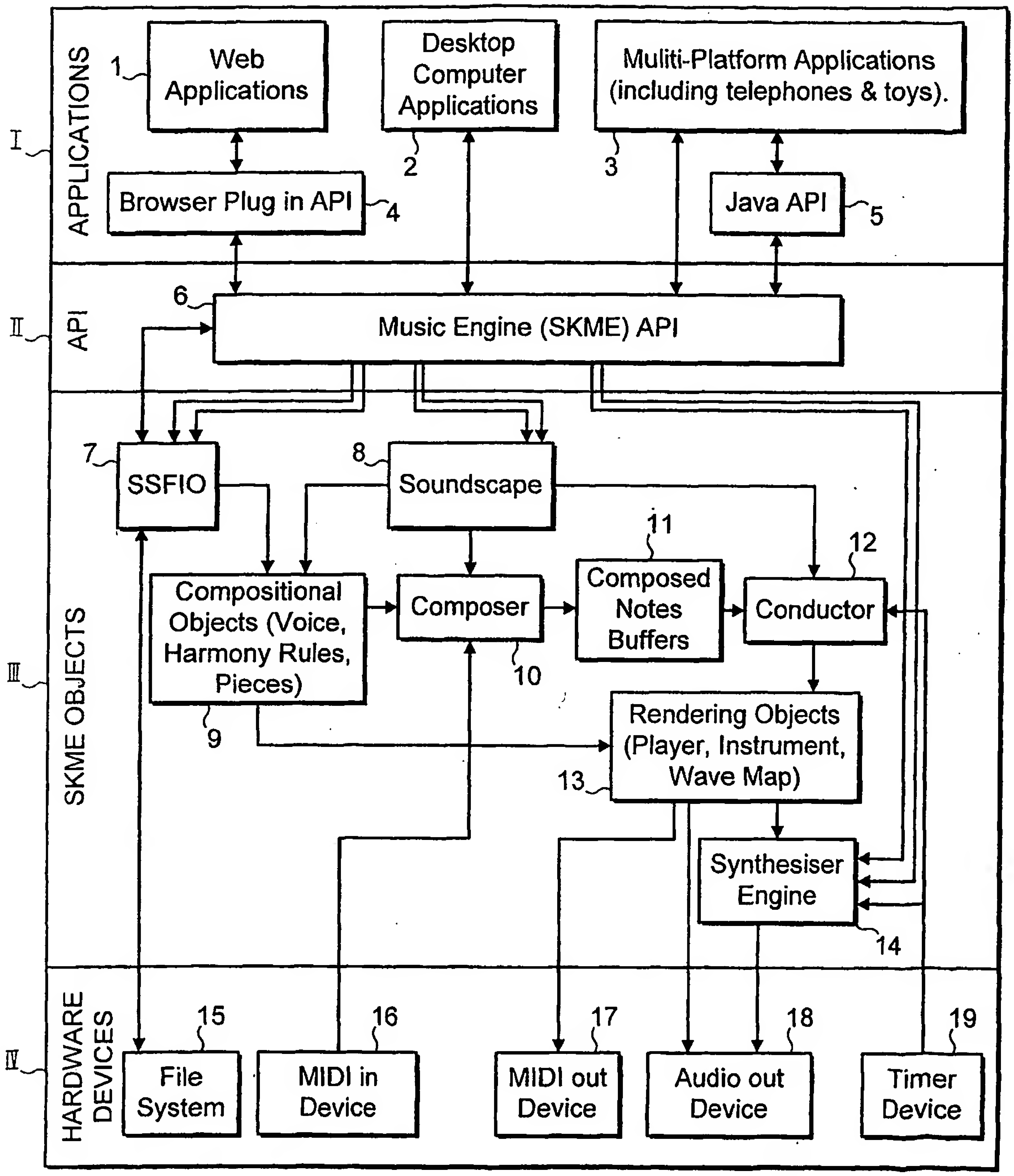
15. A computer including a generative sound system as claimed in any one of the preceding claims.

16. A computer game including a generative sound system as claimed in any one of claims 1 to 14.

17. A mobile phone including a generative sound system as claimed in any one of claims 1 to 14.

18. A generative audio system as claimed in any one of claims 1 to 10 in which the trigger is representative of a speech event.

19. A generative audio system as claimed in claim 10 in which the trigger is generated by the generative audio system itself.



APPLICATIONS
I
1
Web Applications
Desktop Computer Applications
Muliti-Platform Applications (including telephones & toys).
2
3
Browser Plug in API
4
Java API
5
API
II
6
Music Engine (SKME) API
SKME OBJECTS
III
7
SSFIO
8
Soundscape
11
12
Compositional Objects (Voice, Harmony Rules, Pieces)
Composer
Composed Notes Buffers
Conductor
10
9
Rendering Objects (Player, Instrument, Wave Map)
13
Synthesiser Engine
14
HARDWARE DEVICES
IV
15
16
17
18
19
File System
MIDI in Device
MIDI out Device
Audio out Device
Timer Device

FIG. 1

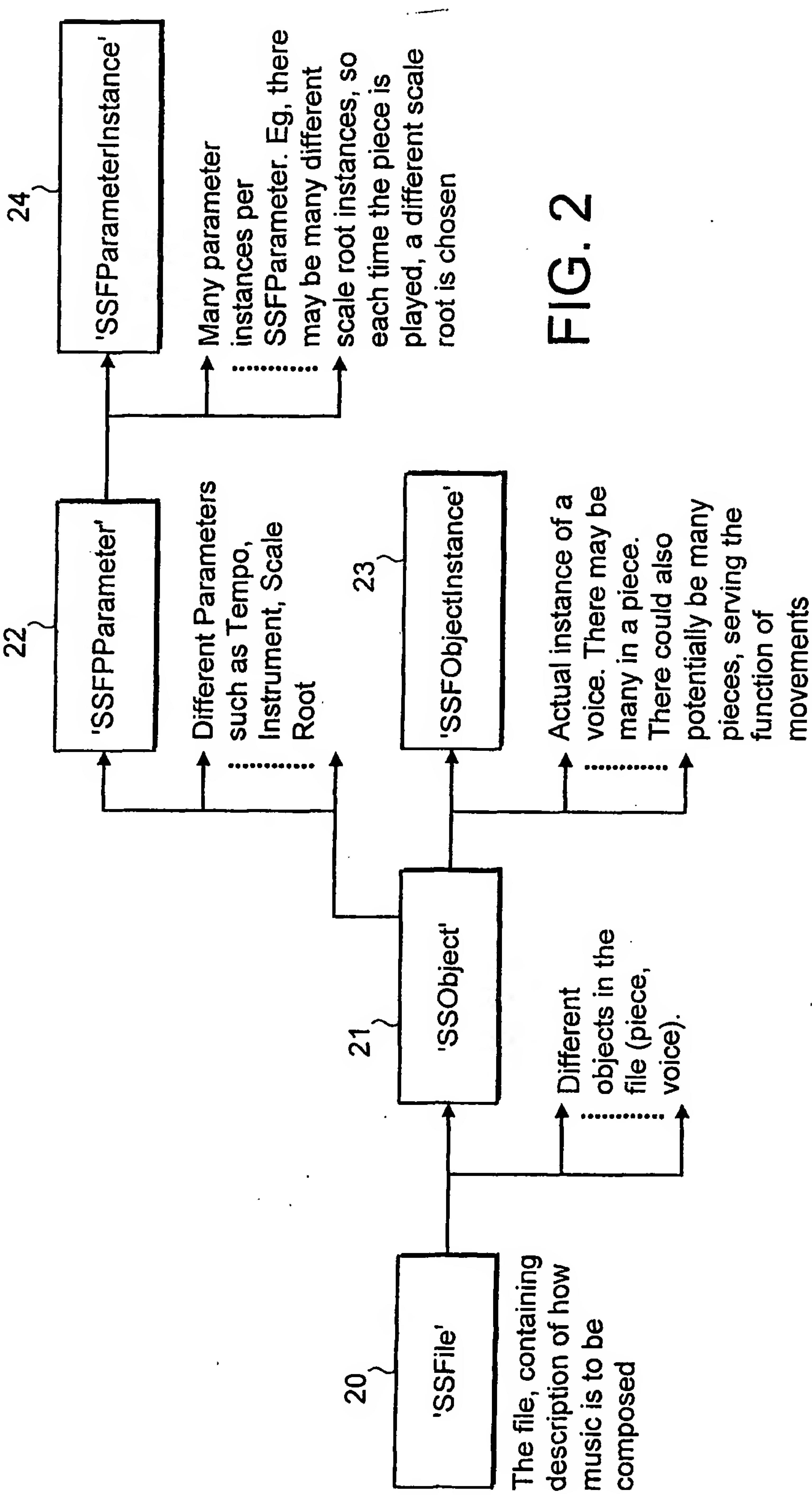

20
'SSFile'
The file, containing description of how music is to be composed
21
'SSObject'
Different objects in the file (piece, voice).
22
'SSFParameter'
Different Parameters such as Tempo, Instrument, Scale Root
23
'SSFObjectInstance'
Actual instance of a voice. There may be many in a piece. There could also potentially be many pieces, serving the function of movements
24
'SSFParameterInstance'
Many parameter instances per SSFParameter. Eg, there may be many different scale root instances, so each time the piece is played, a different scale root is chosen

FIG. 2

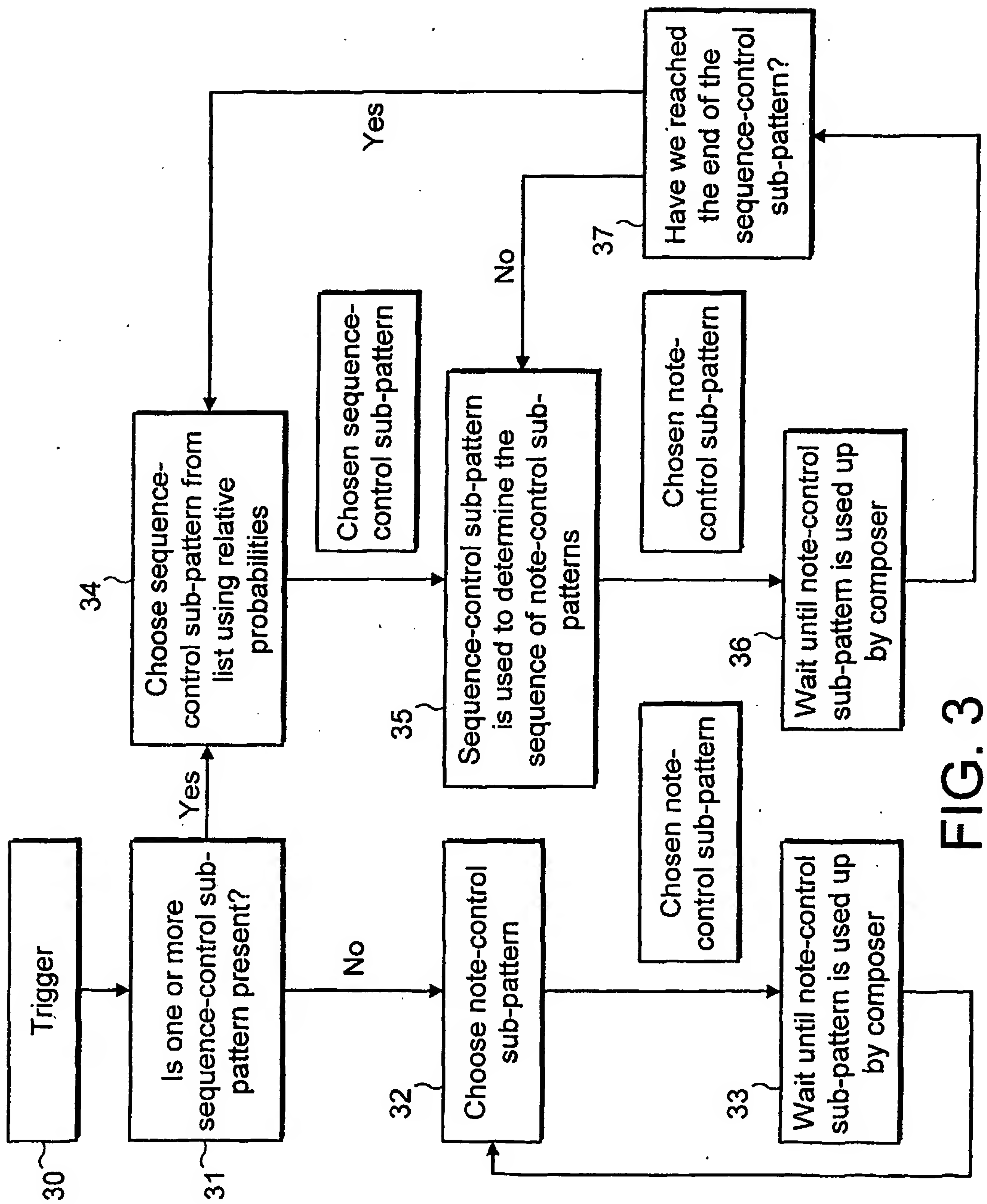
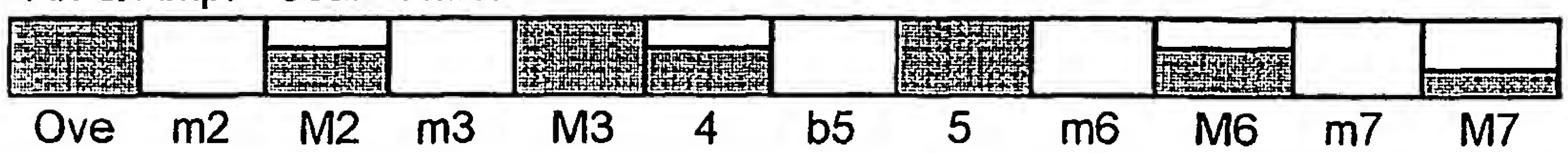

30
Trigger
31
Is one or more sequence-control sub-pattern present?
Yes
No
34
Choose sequence-control sub-pattern from list using relative probabilities
Chosen sequence-control sub-pattern
35
Sequence-control sub-pattern is used to determine the sequence of note-control sub-patterns
Chosen note-control sub-pattern
36
Wait until note-control sub-pattern is used up by composer
37
Have we reached the end of the sequence-control sub-pattern?
Yes
No
32
Choose note-control sub-pattern
Chosen note-control sub-pattern
33
Wait until note-control sub-pattern is used up by composer

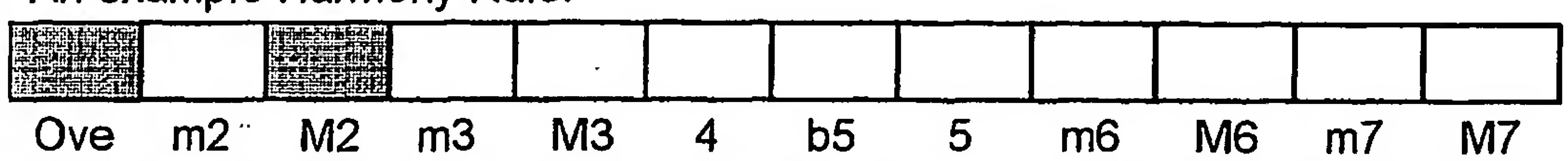
FIG. 3

1 An example Scale Rule:



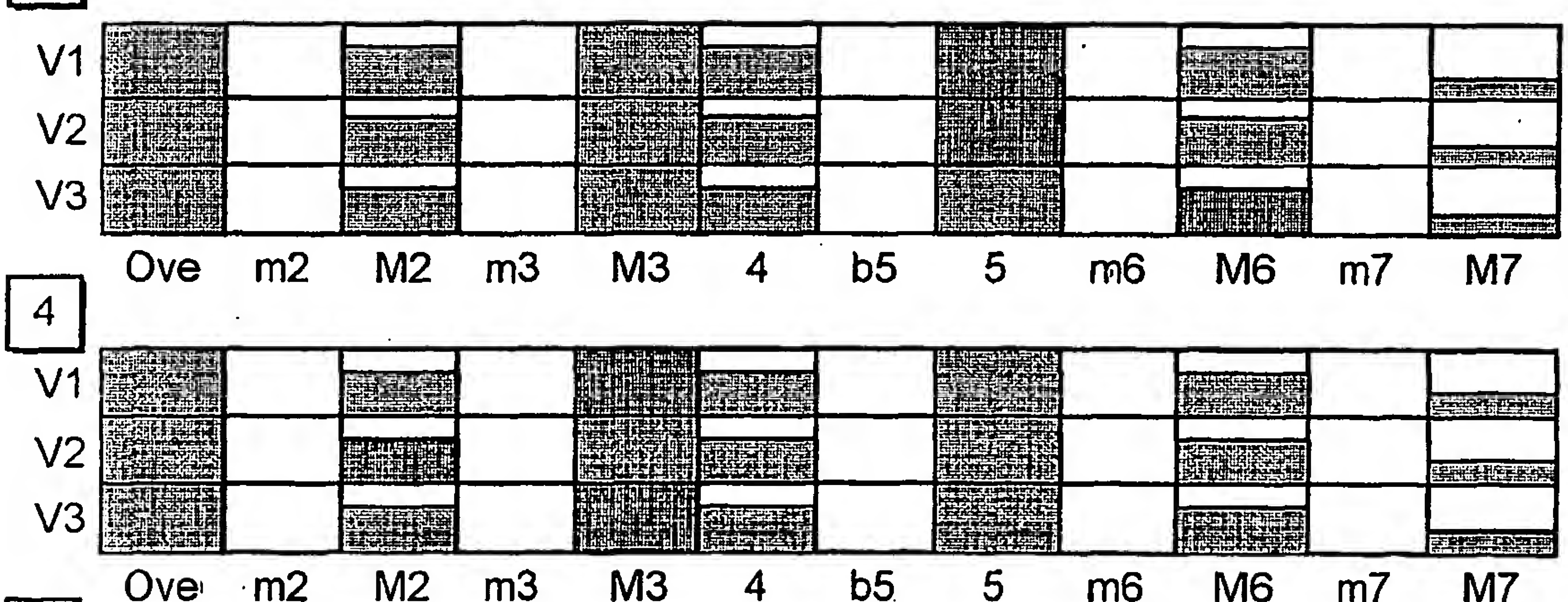
Relative to a scale root, a sequence of notes chosen from this scale rule might be:
Ove, M3, Ove, M2, 5, M6, M3, M2, Ove, 5, M3, M7, Ove, 4
With the scale root set at C, this sequence becomes:
C, E, C, D, G, A, E, D, C, G, E, B, C, F

2 An example Harmony Rule:



3

4



5

Time Sequence:
S:V1:G, S:V2:G, S:V3:A,
E:V1. E:V2, E:V3
S:V2:D, S:V3:E, S:V1:E
E:V2, E:V3, E:V1

FIG. 4

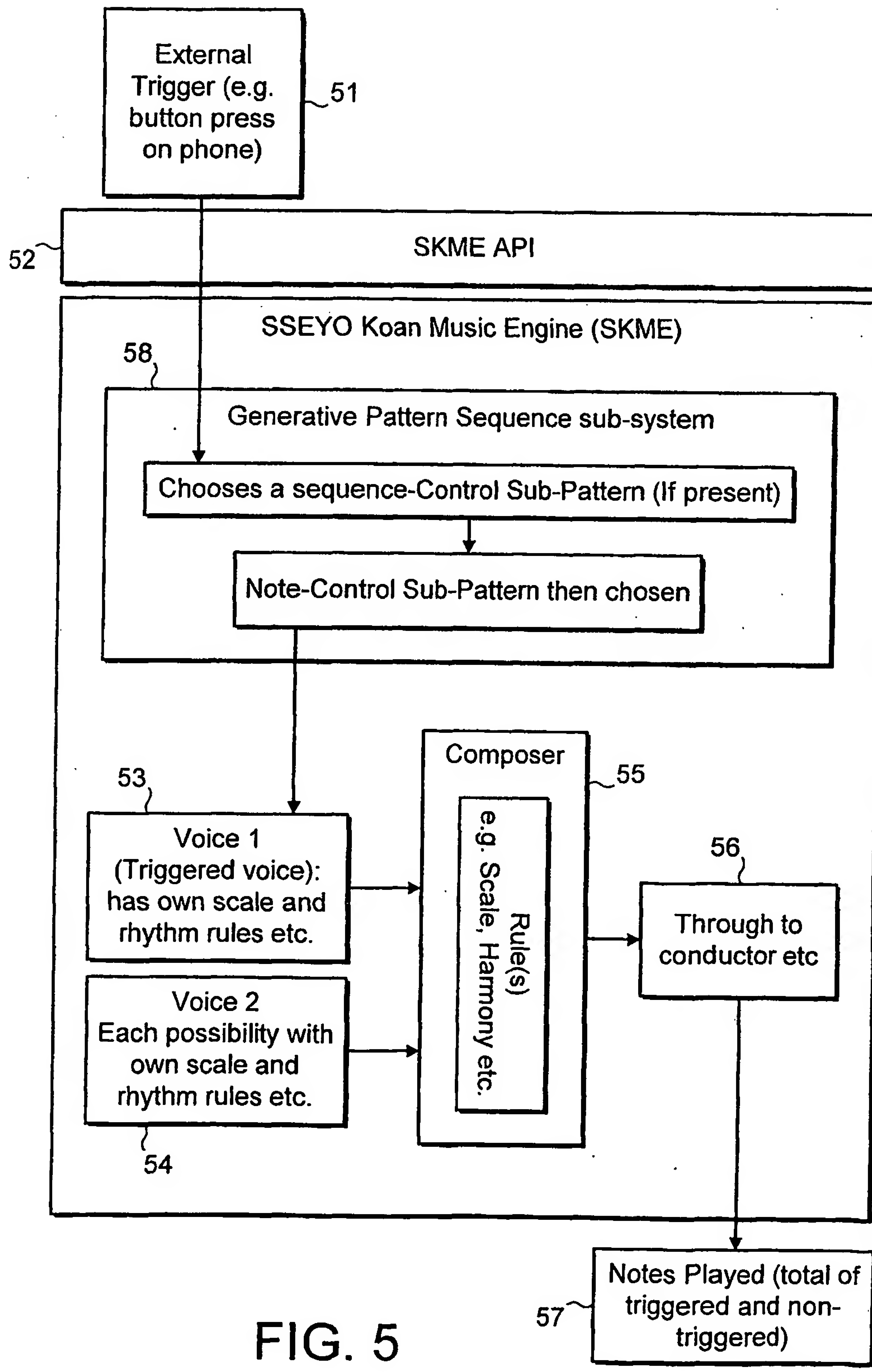

External Trigger (e.g. button press on phone)
51
52
SKME API
SSEYO Koan Music Engine (SKME)
58
Generative Pattern Sequence sub-system
Chooses a sequence-Control Sub-Pattern (If present)
Note-Control Sub-Pattern then chosen
53
Voice 1 (Triggered voice): has own scale and rhythm rules etc.
Voice 2 Each possibility with own scale and rhythm rules etc.
54
Composer
55
Rule(s) e.g. Scale, Harmony etc.
56
Through to conductor etc
57
Notes Played (total of triggered and non-triggered)

FIG. 5